EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Análisis comparativo de lenguajes de programación para el desarrollo de aplicaciones en Ciencia de Datos

presentada por

Lic. Ricardo Gudiel López Pérez

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Joaquín Pérez Ortega

Cuernavaca, Morelos, México. Febrero de 2020.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 19/diciembre/2019

OFICIO No. DCC/175/2019
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

C. DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **C. Lic. Ricardo Gudiel López Pérez**, con número de control M17CE094, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado "**Análisis comparativo de lenguajes de programación para el desarrollo de aplicaciones en Ciencia de Datos**" y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

Dr. Joaquín Pérez Ortega
Doctor en Ciencias Computacionales
4795984
Director de tesis

Dr. José Crispín Zavala Díaz
Doctor en Ciencias Computacionales
3406871
Revisor 1

Dr. Dante Mújica Vargas
Doctor en Comunicaciones y Electrónica
09131756
Revisor 2

Dr. José María Rodríguez Lelis
Doctor en Ciencias en Ingeniería Mecánica
4500026
Revisor 3

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante
JGGS/lmz

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación y Desarrollo Tecnológico
Subdirección Académica

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

Cuernavaca, Morelos, 31/enero /2020

OFICIO No. SAC/102/2020
Asunto: Autorización de impresión de tesis

LIC. RICARDO GUDIEL LÓPEZ PÉREZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS
DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado "***Análisis comparativo de lenguajes de programación para el desarrollo de aplicaciones en Ciencia de Datos***", ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

SEP TecNM
CENTRO NACIONAL
DE INVESTIGACIÓN
Y DESARROLLO
TECNOLÓGICO
SUBDIRECCIÓN
ACADÉMICA

C.p. M.E. Guadalupe Garrido Rivera. Jefa del Departamento de Servicios Escolares.
Expediente.

GVGR/CHG

Dedicatoria

A mis compañeros y amigos: Santiago, Juan Carlos, Violeta, Iván, Xicotencatl, Heidi, Andrea, Celia, Daniel, Dersein, Israel y Lili que fueron indispensables en mi etapa formativa.

A la familia Soriano Santos y Santos Victoriano, que estuvieron conmigo en estos años de estudio, de ellos siempre recibí buenas atenciones.

Y a todos aquellos que directa o indirectamente participaron en mi formación académica.

Agradecimientos

¡Oh abismo de riqueza, de sabiduría y de ciencia el de Dios! ¡Cuán insondables son sus designios e inescrutables sus caminos!
Romanos 11:33

Agradezco a Dios por la vida y por la salud. Por la oportunidad de alcanzar una meta más. A CONACYT por el apoyo económico que hizo posible la realización de la maestría. Al Tecnológico Nacional de México y al Centro Nacional de Investigación y Desarrollo Tecnológico, por ser las instituciones que hicieron posible este logro.

Al Dr. Joaquín Pérez Ortega por ser el director de esta tesis y a cada uno de los miembros del comité tutorial: Dr. José Crispín Zavala Díaz, Dr. José María Rodríguez Lelis y Dr. Dante Mújica Vargas; por cada una de sus observaciones, sugerencias y consejos a lo largo de mis estudios de maestría. A la Dra. Leticia Sánchez Lima, por su apoyo y asesoría en la redacción de este documento de tesis, sus consejos y observaciones fueron de gran ayuda, gracias por su tiempo.

A mis padres, por ser mi apoyo siempre incondicional.

Resumen

En esta investigación se aborda el problema de seleccionar los lenguajes de programación más apropiados para el desarrollo de aplicaciones en Ciencia de Datos. Esto es un problema debido a que en la actualidad existen muchos lenguajes de programación utilizados en varios ámbitos y en distintas aplicaciones, por ello, es necesario identificar los lenguajes que tengan las características más favorables que permitan el desarrollo de aplicaciones en el área de Ciencia de Datos.

Para resolver esta problemática, se siguió un enfoque de solución que se desarrolló en cuatro etapas. En la primera, se seleccionaron dentro de un conjunto de lenguajes de programación, los más utilizados en esta área de conocimiento. En la segunda etapa, se caracterizaron los lenguajes elegidos, los cuales fueron Python y R, en esta etapa, se conocieron los rasgos distintivos de cada lenguaje. En la tercera etapa, se estudiaron de manera teórica y práctica. En la última etapa, se desarrolló un conjunto de casos de prueba para comparar los resultados de la función K-means, que se trata de un algoritmo utilizado en la técnica del agrupamiento de datos, asimismo, este último es utilizado en el ámbito de Ciencia de Datos.

Al experimentar con 10 bases de datos y 30 muestras tomadas de una de las bases de datos llamada *3D_spatial_network* se observó que R presentó un promedio en tiempo de ejecución 1.373487697 segundos y Python 1.737771034 segundos, lo que muestra que R en esta experimentación resultó 1.26 veces más veloz que Python, R mostró una tendencia de ser más veloz que Python, esto se observó debido a que en 38 ejecuciones de 40, obtuvo ventaja en velocidad. En calidad de agrupamiento, para esta experimentación, Python logró en 18 ocasiones mejor calidad y R en 22 ocasiones. En promedio Python obtuvo una pérdida de calidad de 0.208982961 % y R 1.124429758 %.

Dos conclusiones relevantes son las siguientes: el lenguaje Python cuenta con una representación de resultados numéricos más precisos que R. Ambos lenguajes tienen funciones para importar archivos de datos de cualquier tamaño, siendo su limitante el tamaño de la memoria virtual del sistema operativo.

Abstract

This research is about the problem of selecting the most selected programming languages for the development of applications in Data Science. This is a problem because there are currently many programming languages used in various scopes and in different applications, therefore, it is necessary to identify the languages that have the most favorable characteristics for the development of applications in the area of Data Science.

The solution approach consisted in four stages. In the first one, was selected the most used programming languages in this area of knowledge. In the second stage the chosen languages were characterized, which were Python and R, at this stage the distinctive aspects of each language were known. In the third stage they were studied in a theoretical and practical way.In the last stage, a set of test cases will be analyzed to compare the results of the K-means function, it is an algorithm used in the technique of data grouping, the latter is used in the field of Data Science.

Experimenting with 10 databases and 30 samples taken from one of the databases called *3D_spatial_network*. It was observed that R presented an average at run time 1,373487697 seconds and Python 1,737771034 seconds, which shows that R in this experimentation was 1.26 times faster than Python, R showed a tendency to be faster than Python, this was observed due to that in 38 executions of 40, it obtained advantage in speed. In grouping quality, for this experimentation, Python had in 18 occasions better quality and R in 22 occasions. On average, Python obtained a loss of quality of 0.208982961 % and R 1,14429758%.

Two relevant conclusions are the following: the Python language has a representation of numerical results more accurate than R. Both languages have functions to import data files of any size, the size of the virtual memory of the operating system being limiting.

Contenido Página

Lista de Tablas **Página**

Lista de Figuras **Página**

Capítulo 1

Introducción

En algún lugar, algo increíble está esperando ser conocido.
Carl Sagan

1.1. Contexto de la investigación

En estas publicaciones [1, 2, 3, 4] se mencionan que actualmente se generan grandes cantidades de datos por distintas instituciones, organizaciones o empresas públicas y privadas dentro de las siguientes áreas: ingeniería, medicina, finanzas, educación, transporte, entre otras. Los dispositivos electrónicos, softwares, sensores, alarmas, y bases de datos son algunas de las fuentes a través de las cuales se genera este gran cúmulo de información.

Los datos se guardan dentro de unidades de almacenamiento que se han desarrollado gracias al avance de la ciencia y tecnología. En este contexto, surgió un área importante entre los conocimientos de la computación, la cual se conoce como Ciencia de Datos. Esta área reúne un conjunto de principios fundamentales que apoyan y guían la extracción de información y el conocimiento de los datos [5, 6].

La Ciencia de Datos, hace posible que personas, empresas o institutos y entre otras organizaciones, tomen decisiones importantes sobre asuntos y hechos relevantes que les interesen resolver o descubrir. Por mencionar un ejemplo, se señala en [4] la aportación que hizo *googl*e durante la epidemia de gripe porcina en el año 2009 para rastrear el progreso de la epidemia con base en el análisis de búsquedas de temas relacionados con dicha enfermedad.

El científico de datos es una persona importante actualmente dentro de las organizaciones. Es un profesional destacado con la formación y la curiosidad para hacer descubrimientos en el área de *Big Data*. Su aparición reciente en la escena comercial se refleja en el hecho de que las empresas ahora están luchando con información que viene en variedades y volúmenes nunca antes encontrados [7].

Actualmente, existe cada vez mayor número de instituciones que impulsan programas de posgrado en Ciencia de Datos. Esto es un aliciente para que, de igual manera, en el Centro Nacional de Investigación y Desarrollo Tecnológico

(CENIDET) se realicen investigaciones sobre esta área de conocimiento. Esta investigación forma parte de una serie de tesis orientadas a consolidar una plataforma para el diseño y desarrollo de aplicaciones en Ciencia de Datos. Para trabajar con ella, se necesitan habilidades matemáticas, estadísticas, analíticas, conocimiento en el manejo de bases de datos, procesamiento, modelado y visualización de información, así como experiencias sobre lenguajes de programación que harán factible el desarrollo de aplicaciones en esta área de conocimiento. Es por este motivo, que conocer de éstos lenguajes, se torna relevante.

1.2. Descripción del problema

Los lenguajes de programación son utilizados para desarrollar distintas aplicaciones en distintos campos de conocimiento, en el ámbito de Ciencia de Datos son importantes para la lectura, la exploración, la realización de modelos y la visualización de datos. Por ello, no todos resultan idóneos para el desarrollo de aplicaciones en este campo de conocimientos. Por lo que es necesario buscar, seleccionar y experimentar con los diferentes lenguajes de programación, aquellos que contengan las características más favorables para el desarrollo de aplicaciones en Ciencia de Datos.

1.3. Objetivo general

Seleccionar al menos dos lenguajes de programación que reúnan los requisitos necesarios para el desarrollo de aplicaciones en Ciencia de Datos mediante un análisis comparativo entre los lenguajes más usados en esta área.

1.4. Objetivos específicos

1. Seleccionar dos de los lenguajes de programación más utilizados en el área de Ciencia de Datos.

2. Recopilar información sobre las características de los lenguajes seleccionados.
3. Realizar un estudio teórico-práctico de los lenguajes seleccionados.
4. Probar experimentalmente con los lenguajes seleccionados para evaluar la factibilidad para el desarrollo de aplicaciones en Ciencia de Datos.

1.5. Alcances y limitaciones de la investigación

Las limitaciones son las siguientes:

1) La implementación computacional del caso práctico se realizará con el equipo y software disponible en el CENIDET.
2) Los lenguajes a seleccionar deberán ser gratuitos o bien que se puedan adquirir mediante gestiones del CENIDET.

Los alcances son los siguientes:

1) La escala, dimensión o tamaño del caso práctico será el mínimo tal que permita mostrar el desarrollo de aplicaciones en Ciencia de Datos usando los lenguajes de programación seleccionados.
2) La validación de los resultados será de manera experimental.

1.6. Enfoque general de solución

En la Figura 1 se muestra el diagrama con el cual se representa el enfoque general de solución propuesto para el desarrollo de esta investigación.

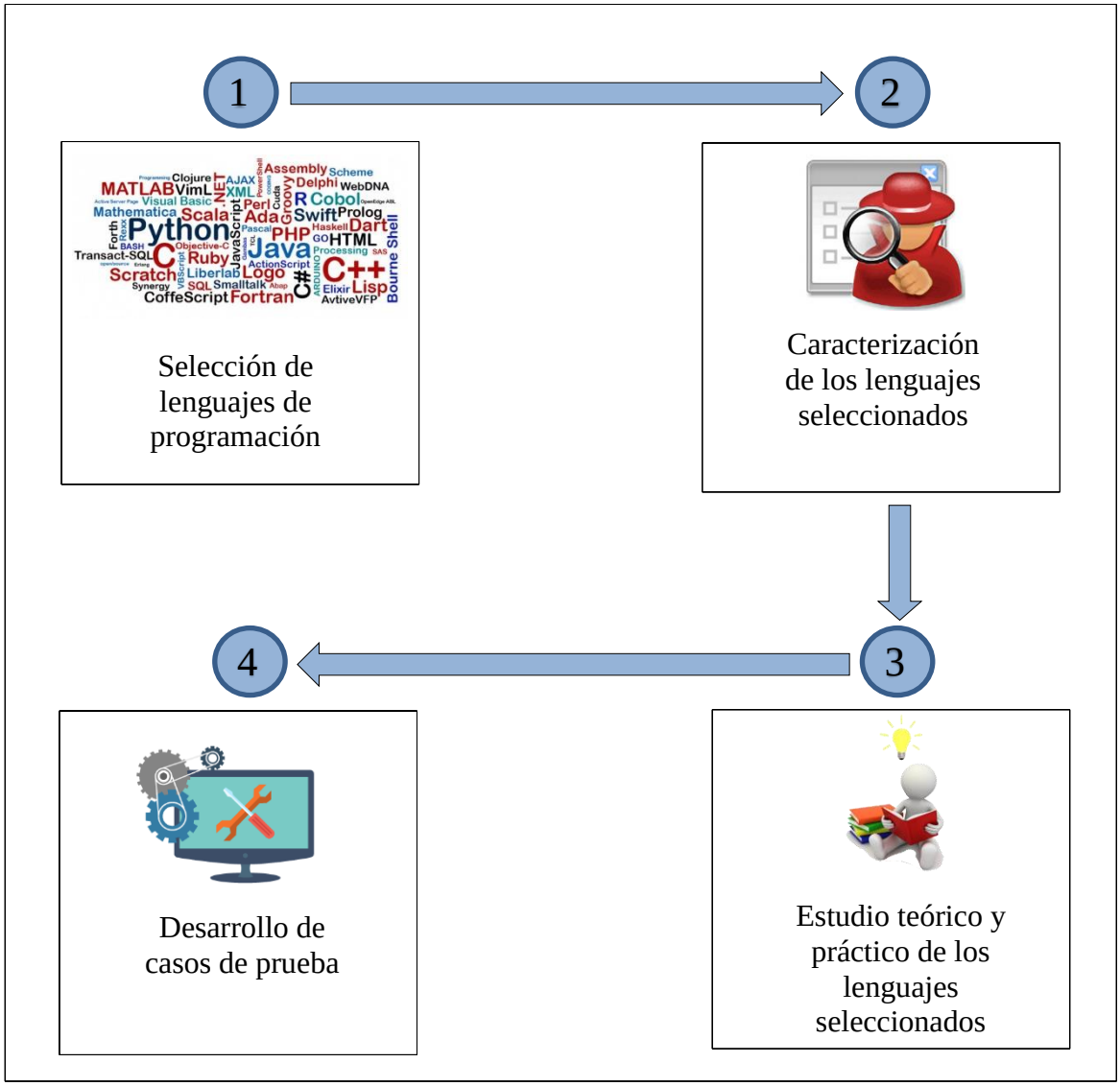

Figura 1 Enfoque general de solución

La investigación se realizó en cuatro etapas. En la primera, se seleccionaron los lenguajes de programación más empleados en el área de desarrollo para aplicaciones en Ciencia de Datos. Esto se hizo con base en criterios que ayudaron a definir cuáles lenguajes son los más utilizados. En la segunda etapa, se caracterizaron los lenguajes elegidos y se conocieron los rasgos distintivos de cada

lenguaje. En la tercera etapa se estudiaron estos lenguajes de manera teórica y práctica apoyándose en la literatura disponible. En la cuarta etapa se desarrolló un conjunto de casos de prueba con el fin de contrastar los resultados que producen los lenguajes elegidos al utilizar la función del algoritmo K-means.

1.7. Organización del documento

Esta tesis consta de 5 Capítulos. En el Capítulo 1, se expone la descripción del problema de investigación que se espera resolver, el contexto de la investigación, objetivos, el enfoque de solución, así como los alcances y limitaciones. En el Capítulo 2, se elabora una introducción teórica en relación con el campo de Ciencia de Datos. Y se incluyen conceptos necesarios dentro de esta área. Asimismo, se señalan conceptos importantes sobre los lenguajes de programación. El Capítulo 3 integra publicaciones acerca de investigaciones en que se han hecho sobre varios análisis comparativos entre diferentes lenguajes de programación. También se incluyen publicaciones especializadas que hacen referencia a los lenguajes de programación más recomendados para el desarrollo de aplicaciones en Ciencia de Datos. El Capítulo 4 hace mención de los criterios tomados en cuenta para la selección de los lenguajes de programación utilizados en Ciencia de Datos, con el objetivo de realizar un análisis comparativo entre estos. En el Capítulo 5 se establecen las conclusiones que se obtuvieron de la investigación y se expresan propuestas de temas para posteriores investigaciones.

Capítulo 2

Marco conceptual

La historia de la ciencia nos demuestra que toda teoría es perecedera. Con cada verdad que nos es revelada ganamos un mejor entendimiento de la naturaleza, y nuestras concepciones y vicisitudes cambian por completo.
Nikola Tesla

2.1. Ciencia de Datos

En 1974, se mencionó por primera vez el concepto de Ciencia de Datos en el libro C*oncise Survey of Computer Methods* para referirse a un área de conocimiento que estudia los datos y la relación entre ellos [8]. En 1996, se mencionó por primera vez el término de Ciencia de Datos en una conferencia de la Federación Internacional de Sociedades de Clasificación (*IFCS*, por sus siglas en inglés) titulada *Ciencia de Datos, clasificación y métodos* [8]. El creciente interés sobre esta área produjo que en el 2002 se estableciera la revista *Data Science Journal,* la primera en su clase [8].

Como se menciona en [9], una definición de Ciencia de Datos desde una perspectiva de abstracción en alto nivel, es la siguiente: *Ciencia de datos es el estudio del dato*. Sin embargo, desde una perspectiva más detallada, Ciencia de Datos es un campo multidisciplinario que integra estadística, informática, computación, comunicación, administración y áreas relacionadas al tratamiento y modelado de la información, tales como minería de datos y aprendizaje automático. De acuerdo con [10] los principales objetivos de este campo de conocimiento son:

a) Facilitar la toma de decisiones basadas en los datos.
b) Descubrir determinados patrones de repetición.
c) Ayudar a predecir comportamientos futuros.
d) Comprender el pasado y también el presente.
e) Apoyar para la creación de nuevas industrias o productos basados en el análisis de los datos.

La Ciencia de Datos puede aplicarse en las siguientes áreas: ingeniería, medicina, finanzas, educación, transporte, salud, política, negocios, comercio electrónico, turismo, entre otros.

2.1.1. Conceptos relacionados con Ciencia de Datos

En la Tabla 1 se explican los conceptos que están relacionados y permiten comprender el campo de la Ciencia de Datos [9].

Tabla 1 Conceptos en el ámbito de Ciencia de Datos

Concepto	Descripción
Análisis de datos	Es el procesamiento de datos con el apoyo de teorías, tecnologías y herramientas tradicionales (por ejemplo, estadística y matemáticas).
Big data	Se refiere a las grandes cantidades de datos que no se pueden manejar con eficiencia y efectividad, por medio de tecnologías, herramientas y esquemas tradicionales de bases de datos.
Científico de datos	Persona que se centra en el manejo, análisis y procesamiento de grandes cantidades de datos.
Análisis descriptivo	Es un tipo de análisis de datos, utilizado en estadística para describir los datos y obtener información.
Análisis predictivo	Es el proceso de predecir eventos futuros y revelar sus razones; mediante teorías, tecnologías, herramientas y procesos que permiten un conocimiento profundo, aunado al descubrimiento detallado de los datos.
Análisis prescriptivo	Es un tipo de análisis que permite optimizar y recomendar acciones para una toma inteligente de decisiones.
Análisis profundo	Se refiere al análisis de datos que permite lograr una efectiva comprensión del porqué y cómo sucedieron las cosas, además de los posibles sucesos en el futuro.

2.1.2. Habilidades requeridas para Ciencia de Datos

En [11] se hace referencia a las habilidades necesarias para que un científico de datos, desarrolle investigación en este campo:

a) Administración de datos.
b) Seguridad, privacidad y ética sobre los datos.
c) Gusto por la investigación, formulación de hipótesis, uso de metodologías para el análisis estadístico.
d) Manejo de herramientas para el análisis de datos.
e) Gestión del flujo de datos.

f) Visualización y presentación de resultados.
g) Habilidades de programación.
h) Conocimiento sobre aprendizaje automático e inteligencia artificial.
i) Habilidades sociales e interpersonales.
j) Conocimiento del dominio.
k) Estrategia de negocios.
l) Sistemas distribuidos.

2.2. Lenguaje de programación

Un lenguaje de programación es aquel que consta de una notación creada e implementada para la codificación de instrucciones, de tal manera que sean ejecutadas por una computadora [12, 13].

2.2.1. Sintaxis

La sintaxis de un lenguaje de programación, se refiere a la forma de la escritura, con respecto a las instrucciones de un programa. Es decir, se trata de la construcción formal de expresiones, comandos, declaración de variables, estructuras de control y entre otras implementaciones que permite el lenguaje [13, 14].

2.2.2. Paradigmas de programación

Dentro del área computacional, un paradigma establece la visión de la ejecución de un programa, así como la organización de su código fuente. La elección de un paradigma señala la forma de pensar, reflexionar y modelar los algoritmos para llegar a la solución de algún problema [14]. En la Tabla 2 se mencionan algunos paradigmas relevantes que se utilizan actualmente.

Tabla 2 Paradigmas de programación

Paradigma	Descripción
Imperativo	Es un modelo de programación que se basa sobre la ejecución de comandos. Estos actualizan los valores de las variables. Un concepto clave de este paradigma es el **procedimiento,** también llamado función**,** el cuál se trata de un conjunto de órdenes que pueden invocarse cuando sea necesario [13, 14].
Orientado a objetos	Su componente principal es el objeto, el cual se define como una entidad abstracta o del mundo real que posee características y métodos. Se relaciona estrechamente con otros conceptos, como los siguientes: clases, subclases, herencia y polimorfismo [13].
Programación lógica	Los datos son los principales actores en este modelo. La idea consiste en definir una aplicación mediante reglas lógicas aplicables a los datos mediante un motor de inferencia [14].
Funcional	Permite centrarse en el análisis sobre los datos y no en los algoritmos que los manipulan. Lo primordial es trabajar con funciones [13].
Scripting	Es una forma de programar basada en el script, este se trata del elemento que se encarga de unir varios subsistemas, de tal manera que, formen un solo sistema. Cada subsistema es un programa diseñado para ser independiente, pero al mismo tiempo debe ser factible para la integración con otros [13].
Secuencial	La programación secuencial consta de una serie de instrucciones que se van ejecutando sucesivamente una tras otra [15].
Paralela	Es un enfoque de programación que se basa en la ejecución de instrucciones entre varios procesadores, de esta manera, se pueden realizar varias tareas de forma simultánea [16].
Heterogénea	Trata de aprovechar las capacidades que ofrecen distintos tipos de procesadores al momento de ejecutar algún programa de cómputo [17].

2.2.3. Sistema del tipo de datos

Los lenguajes de programación utilizan varios tipos de datos. Sin embargo, para entender cómo interactúan los lenguajes con esos diferentes datos, es necesario comprender los conceptos del sistema del tipo de datos. Estos conceptos son: débil vs. fuerte, dinámico vs. estático; mismos que se describen a continuación:

a) Débil vs. fuerte: el sistema o tipado débil únicamente se enfoca en el contenido de la variable, independientemente del tipo de dato que sea. Por otra parte, mientras que un tipado fuerte asigna la misma importancia al contenido y al tipo. Por ejemplo, PHP es un lenguaje débilmente tipado, la cifra 1 y el carácter ‘1’ son idénticos y su comparación mediante el operador de igualdad == devuelve verdadero. En un lenguaje fuertemente tipado, 1 siempre será diferente de ´1´ [14].
b) Estático vs. dinámico: El tipado estático consiste en declarar el tipo de dato de las variables. Por su parte, en el dinámico, el tipo de dato se va actualizando con base en el valor de la variable [13, 14].

Capítulo 3

Estado del arte

Toda ciencia comienza como filosofía y acaba como arte.

William James Durante

3.1. Investigaciones relacionadas

A continuación, se describen varios artículos que están relacionados con el trabajo del análisis comparativo sobre lenguajes de programación en el ámbito del análisis de datos y los campos de estudio relacionados. Los siguientes subtítulos corresponden a cada uno de los títulos de los respectivos artículos.

3.1.1. Estudio comparativo de herramientas computacionales para el agrupamiento en Minería de Datos.

En [18] se estudian siete herramientas computacionales que son utilizadas para el análisis de la información tomando como base la técnica del agrupamiento de datos. Los autores caracterizaron cada una de las herramientas con el fin de conocer sus ventajas y desventajas.

En la Tabla 3 se describen algunas características mencionadas de las herramientas. En la primera columna, se menciona el nombre de la herramienta; la segunda corresponde al nombre del desarrollador; en la tercera el tipo de licencia; en la cuarta columna se menciona el tipo de interfaz que utiliza; en la quinta columna se mencionan los distintos algoritmos de agrupamiento de datos que posee la herramienta; y en la sexta columna, el *website* donde se localiza dicha herramienta.

Tabla 3 Comparación de herramientas que utilizan algoritmos de agrupamiento

Nombre de la herramienta	**Desarrollador**	**Tipo de Licencia**	**Tipo de Interfaz**	**Algoritmos de agrupamiento**	***Website***
Weka	Univerdad de Waikato, en Nueva Zelanda.	Código libre.	Interfaz gráfica y línea de comandos.	Linkage-based, K-means, X-means, Agrupamiento EM, DBSCAN, Optics.	www.cs.waikato.ac.nz /ml/weka
ELK.	Universidad de Munich: Ludwig Maximilian, en Alemania.	Código libre.	Interfaz gráfica y línea de comandos.	Propagación por afinidad, agrupamiento basado en la densidad, agrupamiento jerárquico, K-means, Bi-clustering, agrupamiento SNN, agrupamiento de correlación.	www.dbs.ifi.lmu.de/ research/KDD/ELKI
Orange	Universidad de Ljubljana, en Eslovenia.	Código libre.	Interfaz gráfica.	Linkage-based, mapeo auto-organizado, K-medoids, fuzzy c-means	http://orange.biolab.si/
R	Laboratorios Bell.	Código libre.	Interfaz gráfica y línea de comandos.	Linkage-based.	http://cran.r-project.org
KNIME	Compañía KNIME.com AG.	Código Libre.	Interfaz gráfica.	Linkage-based, K-means, Fuzzy c-means, X-means, agrupamiento EM, DBSCAN, agrupamiento ANN, Optics.	http://www.knime.org/
Scikit-learn	Múltiples desarrolladores de INRIA, telecom Paris Tech y Google.	Código Libre.	Línea de comandos.	K-means, Propagación por afinidad, Mean Shift, Spectral, Jerarquía Ward, agrupamiento aglomerativo, DBSCAN, Mezclas gaussianas, Birch.	http://scikit-learn.org/
RapidMiner	Compañía Rapid Miner en Alemania.	Propiedad privada (versión 6.1).	Interfaz gráfica.	Linkage-based, K.means, X-means, agrupamiento EM, DBSCAN, mapa auto-organizado ANN, Optics.	https://rapidminer.com/

Se observa que en varias de las herramientas mencionadas se pueden trabajar bajo una licencia gratuita, únicamente *RapidMiner* es de propiedad privada. Por otro lado, a excepción de *Scikit-learn,* todas tienen una interfaz gráfica, lo que facilita su uso. Como se puede notar, varias de estas herramientas utilizan distintos algoritmos de agrupamiento, algunos de los más frecuentes son: K-means, DBSCAN, *Linkage-based* y X-means. Es destacable que tres herramientas fueron desarrolladas por institutos universitarios, esto quiere decir que el desarrollo de software para realizar análisis de datos no está limitado a empresas privadas, sino que el medio académico puede realizar contribuciones importantes en la materia.

3.1.2. Matlab vs. Python vs. R

En [19] se menciona que dentro del contexto educativo universitario, específicamente dentro de las áreas de matemáticas y estadística se realizó la comparación entre los siguientes lenguajes de programación; lenguajes que fueron usados satisfactoriamente para impartir la enseñanza en dichas materias: Matlab, Python y R. Con base en las características de cada lenguaje y contrastando las ventajas y desventajas de cada uno de ellos, se determinó que Python es el lenguaje de programación recomendable para la enseñanza. Sin embargo, R es potencialmente una opción recomendable para usarse en materia de minería de datos. También se mencionó que R, SPSS y SAS son herramientas de software factibles para su uso en el análisis de datos dentro del contexto de *Big Data*.

3.1.3. Análisis comparativo de eficiencia de lenguajes de programación

En esta investigación [20], se realizó una comparación de la eficiencia de diferentes lenguajes para ejecutar el algoritmo llamado Prob2_SUM. Descrito de manera sucinta, trata de determinar si un número cualquiera está constituido por la suma de 2 términos. Ambos elementos son miembros de un conjunto acotado de valores. La tarea se ejecutó en la misma plataforma (equipo, sistema operativo). Asimismo se evaluaron variantes de distintas máquinas virtuales y versiones para un mismo lenguaje, los cuales fueron los siguientes: Java NetBeans 7.3.1 JDK 1.6 JVM jrockit;

Java NetBeans 7.3.1 JDK 1.7 JVM default; C++ Microsoft Visual Studio .NET 2003; C++ NetBeans 3.7.1; Python PyCharm 2.7.1 python.exe; Python PyCharm 2.7.1 pypy.exe y Smalltalk Squeak-4.3.

Con esta investigación se concluyó que los lenguajes eficientes en orden de importancia de mayor a menor fueron: Java, C++ y Python con PyPy. En materia de eficacia fueron los siguientes: Smalltalk y Python interpretado. Estos dos últimos son capaces de resolver el problema, pero no consideran el tiempo como un factor importante.

3.1.4. Comparación de lenguajes de programación en macroeconomía

En esta investigación [21], los autores implementaron el modelo de crecimiento neoclásico estocásico en el área de economía usando los siguientes lenguajes de programación: C++, Fortran, Java, Julia, Python, Matlab, Mathematica y R. Su objetivo fue medir los tiempos de ejecución en cada uno de ellos y orientar a los economistas sobre qué lenguaje de programación podrían usar de acuerdo con sus necesidades. Cuando las ejecuciones tardaban menos de 60 segundos, lo repetían 10 veces y obtenían el tiempo promedio, esto se realizó para aminorar las diferencias causadas por el sistema operativo. Pero si la ejecución tardaba más de un minuto, solo se realizaba una ejecución, porque consideraron que no impactaba de manera importante sobre el tiempo.

Se determinó que C++ y Fortran siguen manteniendo ventaja sobre los otros lenguajes en materia de velocidad. Por ejemplo, Java es 2.10 a 2.69 veces más lento que C++; Matlab aproximadamente 10 veces más tardado; y la implementación PyPy de Python alrededor de 48 veces más lento. Por otro lado, Fortran es un lenguaje considerado como simple y fácil de aprender, lo cual es una ventaja cuando existe código legado. También se obtuvo que Julia es de 2.64 a 2.70 veces más lento que C++. Julia es ligeramente más rápido que Java y aproximadamente cuatro veces más rápido que Matlab. En la implementación

tradicional de Python, se ejecutó entre 155 y 269 veces más lento que C++; R es 475 a 491 veces más lento que C++. Aunque, con la versión compilada de R, mejoró su desempeño y fue de 243 a 281 veces más lento. Mathematica fue 809 veces más lento que C++.

3.1.5. Estudio comparativo de lenguajes de programación con la plataforma *Rosetta Code*

En [22] se realizó un análisis comparativo de los lenguajes más importantes dentro de los distintos paradigmas de programación: funcional, orientado a objetos, procedural e interpretados. En el ámbito de lenguajes funcionales se tomaron en cuenta: F# y Haskell. Para el paradigma orientado a objetos se consideraron: C# y Java. En los lenguajes procedurales fueron: C y Go. Por último, en el paradigma de lenguajes interpretados se seleccionaron los siguientes: Python y Ruby. El análisis se realizó con el apoyo del repositorio *Rossetta Code*, que contiene 7,087 soluciones de 745 tareas.

Se observó que los lenguajes interpretados y funcionales fueron más concisos que los orientados a objetos y procedurales. Los compiladores de Go y Haskell produjeron tamaños de archivos ejecutables mayores que los que se compilan en *bytecode*, los cuales fueron: F#, C#, Java y Python. Al medir los desempeños en tiempo de ejecución, resultó que C fue el mejor sobre los demás lenguajes, por ejemplo fue 1.6 veces más rápido que Go, el cual quedó en segundo lugar. Con respecto al uso eficiente de memoria RAM, los dos más sobresalientes fueron C y Go. Go usó en promedio 1.8 veces más memoria que C. Los demás lenguajes utilizaron en promedio de 2.5 a 14.2 veces más memoria en comparación con estos dos lenguajes mencionados anteriormente. Por último, se mencionó que Go fue el menos propenso a presentar errores en tiempo de compilación con respecto a los demás.

3.1.6. Comparación de los lenguajes de programación C, Matlab y Python para la enseñanza en la ingeniería.

El objetivo de este trabajo de investigación [23] fue realizar una comparación en el contexto de la enseñanza y el aprendizaje con respecto a los siguientes lenguajes de programación: C, Matlab y Python. Para compararlos, se realizó el ejercicio de la obtención del área con la regla trapezoidal compuesta con *n* divisiones del intervalo [*a, b*]. Después de realizar dicho ejercicio e implementarlo en los tres lenguajes con estudiantes de primero y segundo grado de la carrera de ingeniería y con estudiantes de posgrado en otras ramas del conocimiento, se analizaron las experiencias y resultados que se obtuvieron para cada lenguaje. A continuación se mencionan dichos hallazgos de manera sucinta:

a) Con respecto a la implementación en el lenguaje C, se menciona que es necesario declarar las variables y sus tipos antes de ejecutar y utilizarlas en las instrucciones. Otra característica, es que el código fuente se tiene que compilar antes de ser ejecutado. Los problemas comunes que presentaron los estudiantes al realizar la codificación fueron los siguientes: la sangría y el alcance para los ciclos *for* y otros bloques de código no estaban relacionados con lo que deberían realizar, se hicieron omisiones del punto y coma, llaves, paréntesis alrededor de la sentencia *if*. Se pasaron tipos de variables incorrectos al momento de llamar una función y también se presentaron errores de impresión al aplicar un formato inadecuado.
b) Al usar Matlab, se observó que en comparación con C, no había que preocuparse por algunos de los problemas anteriormente mencionados, debido a que elementos como las llaves en bloques de código o la declaración de tipos de variables no son aplicados en Matlab. Este lenguaje es capaz de trabajar con distintas funciones, una en cada archivo. Sin embargo, dicha característica se convierte en una desventaja al momento de usar varias funciones.
c) En referencia a Python, en este artículo se expresa que al igual que Matlab tiene la ventaja de no utilizar algunos elementos mencionados anteriormente, como el punto y coma o las llaves, pero que en C son

utilizados obligatoriamente. Una diferencia notable entre Python y Matlab es que este último necesita de la palabra reservada *end* para terminar el ciclo *for*, mientras que en Python no es necesario porque trabaja con sangrías. De esta forma delimita un bloque de código que pertenece a un ciclo. Los estudiantes presentaron mayor afinidad para la codificación de programas utilizando Python sobre los otros lenguajes mencionados en esta investigación.

3.1.7 Revisión de lenguajes de programación y herramientas utilizadas para el análisis en Big Data

En [24] se realizó una comparación de las herramientas usadas en el área de *Big Data*. En la categoría de lenguajes de programación, se tomaron en cuenta los siguientes: Python y R. Para comparar software se consideraron estos: SAS y Sql Server. En la clase de herramientas estadísticas: SPSS, STATA, MiniTab y Statistica. También se compararon los siguientes softwares de visualización: Tableau, Qlikview y Spotfire; por último para la visualización *web* se compararon estas herramientas: D3, Protovis y Flash.

Como conclusión de esta investigación se expresó que la herramienta D3 es dos veces más rápido que Protovist y 3 veces más rápido que Flash para la graficación de datos. R es el lenguaje de programación que suelen usar los científicos de datos. Recomiendan la herramienta SPSS para personas que no son profesionales en la rama estadística. Finalmente, para la utilización de un software de presentación visual exhortan el uso de Tableau.

En la Tabla 4 se establece una comparación entre los estudios analizados en este capítulo. La primeras dos columnas mencionan el número y título de investigación, respectivamente. La tercera columna señala el contexto del trabajo, es decir, el enfoque que se le dio referente a un campo de estudio o ámbito de aplicación. La cuarta columna expresa los lenguajes de programación que se compararon. En la quinta columna se señalan cuáles fueron las características que

se tomaron en cuenta para realizar el análisis. En la última columna se mencionan los resultados que se obtuvieron en cada investigación.

Tabla 4 Investigaciones sobre la comparación de lenguajes de programación

No.	Título	Áreas de aplicación	Lenguajes de programación comparados	Características comparadas	Resultados
1	Estudio comparativo de herramientas computacionales para el agrupamiento en Minería de Datos.	Minería de datos.	R, Scikit-learn (basado en Python).	Tipo de licencia, tipo de interfaz, empresa desarrolladora, algoritmos disponibles para el agrupamiento de datos.	*RapidMiner* es de propiedad privada. Por otro lado, a excepción de *Scikit-learn,* todas tienen una interfaz gráfica,
2	Matlab vs. Python vs. R.	Enseñanza en Matemáticas y Estadística	Matlab, Python, R.	Tipo de licencia, portabilidad en sistemas operativos, herramientas de visualización, demanda laboral para el personal capacitado en esos lenguajes, sintaxis (definición de ciclos).	Se determinó que Python es el lenguaje de programación recomendable para la enseñanza.
3	Análisis comparativo de eficiencia de lenguajes de programación.	Algoritmia	Java, C++, Python, Smalltalk.	Eficiencia en tiempo de ejecución, eficacia.	Lenguajes más eficientes: Java, C++ y Python con PyPy. En materia de eficacia fueron los siguientes: Smalltalk y Python interpretado.
4	Comparación de lenguajes de programación en macroeconomía.	Economía	C++, Fortran, Java, Julia, Python, Matlab, Mathematica, R.	Eficiencia en tiempo de ejecución.	Se determinó que C++ y Fortran siguen manteniendo ventaja sobre los otros lenguajes en materia de velocidad.
5	Estudio comparativo de lenguajes de programación con *Rosetta Code.*	Algoritmia	C, Go, C#, Java, F#, Haskell, Python, Ruby.	Líneas de código, tamaño de archivos ejecutables, eficiencia en tiempo de ejecución, uso de memoria RAM, errores de compilación.	Al medir los desempeños en tiempo de ejecución, resultó que C fue el mejor sobre los demás lenguajes, por ejemplo fue 1.6 veces más rápido que Go, el cual quedó en segundo lugar.
6	Comparación de C, Matlab y Python para la enseñanza en la ingeniería.	Algoritmia y Educación	C, Matlab, Python.	Sintaxis, implementación de funciones.	Los problemas comunes que presentaron los estudiantes al realizar la codificación fueron los siguientes: la sangría y el alcance para los ciclos *for,* sobre todo en el lenguaje C.
7	Revisión de lenguajes de programación y herramientas utilizadas para el análisis en Big Data	Big Data.	Python y R.	Librerías, esfuerzo de aprendizaje, velocidad de ejecución, herramientas de visualización, entornos de desarrollo.	R es el lenguaje de programación que suelen usar los científicos de datos.

Estas investigaciones realizaron trabajos comparativos entre diferentes lenguajes de programación enfocados en diversas áreas de aplicación pero ninguna expone una comparación en el área de Ciencia de Datos. Se observó que Python, R, Matlab, y Java fueron de los lenguajes de programación que con más frecuencia se consideraron para analizarlos y compararlos. Es destacable el trabajo experimental que se realizaron en estas investigaciones, específicamente en el análisis de la eficiencia del tiempo de ejecución entre los diferentes lenguajes. Sin embargo, otros trabajos solamente se limitaron en caracterizarlos de manera teórica, por ejemplo: tipo de licencia, tipo de interfaz, nombre de la empresa desarrolladora, entre otros.

3.2. Lenguajes de programación para desarrollar aplicaciones en Ciencia de Datos

En la Tabla 5 se mencionan las investigaciones publicadas que señalan los lenguajes de programación usados para el desarrollo de aplicaciones en Ciencia de Datos. Particularmente referidas al uso de Python y R.

Tabla 5 Publicaciones que refieren a Python y R como los lenguajes de programación más empleados en Ciencia de Datos

Autor (es)	Título
T.H. Davenport y D. J. Patil.	Data Scientist: The Sexiest Job of the 21st Century, 2012 [7]. Artículo clásico que cuenta con 1158 citas.
D. Donoho.	50 years of Data Science, 2015 [25]. Artículo clásico que cuenta con 134 citas.
R. A. Muenchen.	The Popularity of Data Science Software, 2019 [26].
T. Siddiqui, M. Alkadri, y N. A. Khan.	Review of Programming Languages and Tools for Big Data Analytics, 2017 [24].
S. Cass, y P. Bulusu.	Interactive: The Top Programming Languages, 2018 [27].
L. Cao	Data science: a comprehensive overview, 2017 [9].
B. Marshall	Data science experience for undergraduates, 2017 [28].
A. Parameswaran	Enabling Data Science for the majority, 2019 [29].

S. Kumar, N. Dhanda y A. Pandey	Data Science – Cosmic Infoset Mining, Modeling and Visualization, 2018 [30].
T. Wiktorski, Y. Demchenko y A. Belloum.	Model Curricula for Data Science EDISON Data Science Framework, 2017 [31].

Con base en las publicaciones especializadas, se observa que en años recientes, el conocimiento de Python y R se ha vuelto demandante. En el medio científico, empresarial e incluso académico apuestan por descubrir las ventajas que estos lenguajes ofrecen. En el ámbito de Ciencia de Datos, ambos lenguajes han tomado un papel importante en el desarrollo de aplicaciones por todo lo que conlleva al momento de trabajar con datos, los cuales se presentan en variadas cantidades y de diferentes tipos, desde números, letras, audios, imágenes, entre otros.

Capítulo 4

Análisis comparativo de lenguajes de programación para aplicaciones en Ciencia de Datos

Ciencia sin conciencia no es más que ruina del alma.

Francois Rabelais

4.1. Criterios de selección de lenguajes

En la página oficial de la revista *Institute of Electrical and Electronics Engineers* (IEEE) Spectrum [27], se publicó una clasificación realizada en 2018 para determinar los lenguajes de programación más usados en el desarrollo de aplicaciones de tipo empresarial, de escritorio y científicas. Se observó que entre los primeros seis lenguajes destacaron los siguientes: Python, C++, Java, C, C# y R. En la Tabla 6 se observa esta clasificación. En la primera columna se muestra el orden de importancia de mayor a menor, en la segunda columna el nombre del lenguaje y en la tercera columna los puntos del *ranking*.

Tabla 6 *Ranking* de lenguajes de programación para desarrollar aplicaciones empresariales, de escritorio y científicas

Número de clasificación	Nombre del lenguaje	Puntuación
1	Python	100.0
2	C++	99.7
3	Java	97.5
4	C	96.7
5	C#	89.4
6	R	82.9

Esta clasificación se realizó con base en el número de menciones que fueron identificadas para cada lenguaje en las páginas web que tomó en cuenta IEEE, estas fueron: *Twitter, Stack Overflow, Reddit, Hacker News*. También se tomó en cuenta el número de visitas (*hits*) para cada lenguaje con la ayuda de *Google Search* y el número de búsquedas de información con respecto al lenguaje de programación, por *Google Trends*. Además, se midió el número de proyectos creados para cada lenguaje y el número de proyectos activos, es decir, los proyectos que son editados o reciben mantenimiento en la plataforma de *GitHub*. Se consideró el índice de la demanda laboral de las personas que usan determinados lenguajes de programación para los puestos ofrecidos en los sitios *web* de *CareerBuilder* y *Dice*.

Específicamente, para el área de Ciencia de Datos, la empresa dedicada al tratamiento de datos llamada *Kaggle*, cuyo propietario es *Google*, realizó una encuesta a los profesionales de Ciencia de Datos preguntándoles a 15,222 personas: ¿Cuál era el lenguaje de programación que empleaban regularmente? Como resultado se obtuvo que Python obtuvo una preferencia del 54 % y con respecto a R el 13 %. Tal como se muestra en la Figura 2:

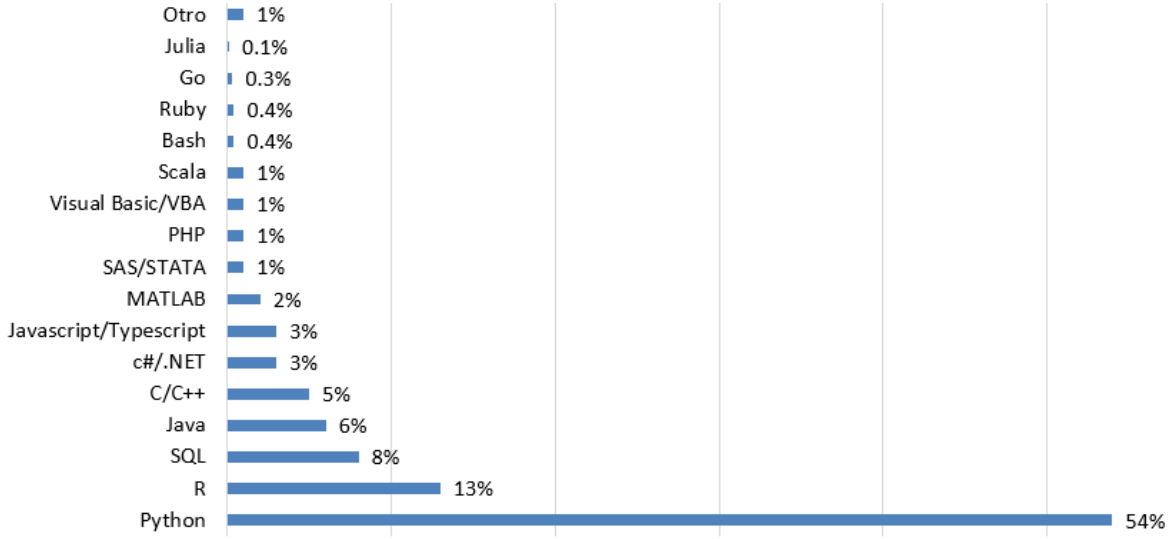

Figura 2 Lenguajes de programación usados en Ciencia de Datos [32]

Otra pregunta realizada a 18,788 científicos de datos, por *Kaggle*, fue la siguiente: ¿Qué lenguaje de programación recomendarías para una persona que quisiera ser un científico de datos? Se obtuvo que el 75 % recomendaría Python y el 12 % R. La Figura 3 muestra los resultados de la encuesta realizada:

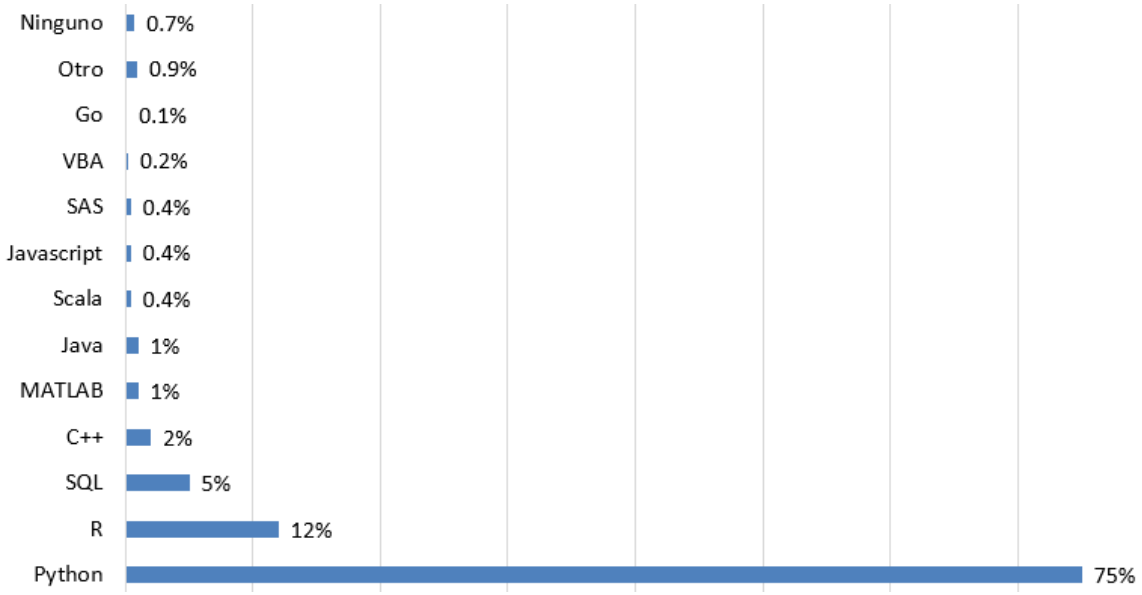

Figura 3 Lenguajes de programación recomendados por profesionales en el área de Ciencia de Datos [32]

KDnuggets, una revista oficial sobre Ciencia de Datos, muestra una investigación de los perfiles de 1,001 científicos de datos en el sitio web *LinkedIn* y se encontró como resultado que 54 % de las personas emplean Python como lenguaje de programación y 45 % R [33]. En la Figura 4 se observa que Python está dominando dentro de las empresas industriales, financieras, tecnológicas y de salud, para el desarrollo de aplicaciones de Ciencia de Datos.

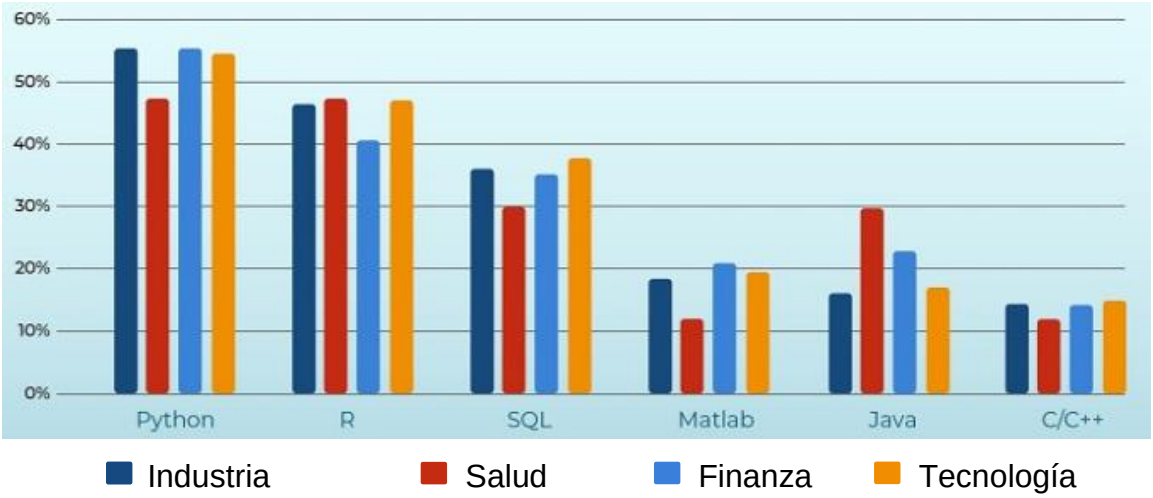

Figura 4 Lenguajes de programación usados en empresas industriales, de salud, financieras y tecnológicas [33]

Por último, como se observa en la Figura 5, *KDnuggets* realizó una encuesta a 2,052 participantes, dedicados a la Ciencia de Datos y al aprendizaje automático, obteniendo. Como resultado se obtuvo que en lo que se refiere a lenguajes de programación, Python y R son los más usados en estas áreas, pero Python ha ido ganando preferencia, tanto así que en el año 2018 se colocó en primer lugar, superando a R.

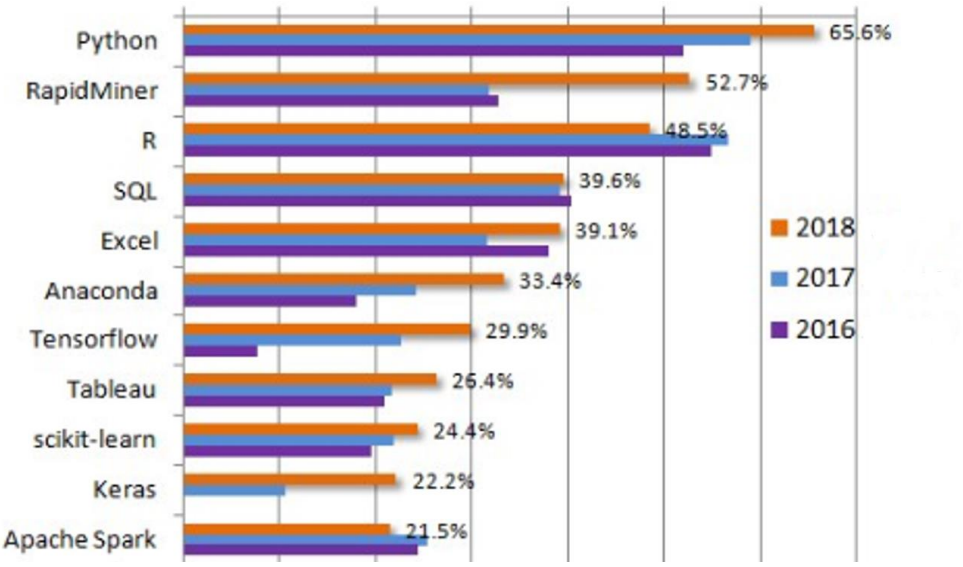

Figura 5 Herramientas para el desarrollo de aplicaciones en Ciencia de Datos [34]

4.2. Selección de los lenguajes

Con base en los cinco criterios de selección: ranking, uso de lenguajes por los científicos de datos, recomendación para el aprendizaje de lenguajes enfocados a Ciencia de Datos, lenguajes de programación utilizados por las empresas y clasificación de herramientas de software, se determinaron los lenguajes de programación más utilizados para el desarrollo de aplicaciones en Ciencia de Datos. Posteriormente, con el fin de realizar un análisis comparativo entre éstos, se eligieron los dos más sobresalientes, los cuales fueron: Python y R. En la literatura investigada y referenciada en el Capítulo 3 de esta tesis, ambos lenguajes fueron mencionados reiterada y enfáticamente por expertos y profesionales en el área de conocimiento en Ciencia de Dato. Frente a esas menciones, no existe incertidumbre

sobre los lenguajes mayormente empleados en este campo multidisciplinario. Con base en la Sección 3.2 de esta tesis, son lenguajes más utilizados durante los últimos siete años, hasta la actualidad.

4.3. Caracterización de los lenguajes seleccionados

En este apartado se describen las características de los lenguajes de programación Python y R, con el objetivo de conocer y especificar el contexto relacionado con ambos lenguajes.

4.3.1. Lenguaje de programación Python

Este es un lenguaje de programación que cuenta con estructuras de datos eficientes y de alto nivel (por ejemplo, vectores, tuplas y *dataframes*). La sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas [35]. Algunos casos de éxito en el uso de este lenguaje son Google, Yahoo, la NASA, Industrias *Light & Magic*, y todas las distribuciones Linux [36].

Python tiene un enfoque de desarrollo orientado a objetos, aunque también tiene la posibilidad de ser imperativo o funcional, lo que hace factible que en algunos casos utilice los tres enfoques a la vez. Su aplicación permite generar documentos de texto, imágenes, flujos de audio o de video, e incluso manipular archivos XML. Hace posible la ejecución de comandos externos, utilizar recursos de la computadora y realizar programación concurrente (tareas y procesos) o de red. Permite comunicarse directamente con dispositivos periféricos como impresoras 3D o servomotores [14].

Las cualidades de Python, son argumentos que permiten que sea una elección prioritaria en el campo de la investigación para desarrollar proyectos diversos y variados. En la historia de la informática existieron lenguajes de programación que han tenido éxito en el mundo empresarial, pero no han penetrado

jamás en el mundo universitario y a la inversa. Python ha conseguido implantarse en ambos medios sin oponerse, gracias a que no se ha diseñado específicamente para uno o para otro y dispone de suficientes librerías que permiten responder a varios tipos de problemáticas. Permite: generar gráficos, hacer cálculo distribuido y es un lenguaje con gran potencial en las matemáticas. Por ejemplo, es un importante recurso para realizar operaciones como las permutaciones [14].

El desempeño de este lenguaje puede ser mejorado mediante el uso de MPI (*Message Passing Interface*) o CUDA (*Compute Unified Device Architecture*). Actualmente, casi todos los sistemas operativos admiten *Python* de tal manera que este lenguaje de programación proporciona portabilidad en muchas plataformas informáticas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS) por lo que si no se utilizan librerías específicas en cada plataforma, el programa podrá ejecutarse en todos estos sistemas sin cambios relevantes [36], [37]. Contiene librerías de funciones para el análisis de datos, entre las cuales se consideran:

a) *Numpy*. Permite realizar la estructura base para los datos con el cual se puede crear un eficiente y multidimensional arreglo de objetos. Contiene una amplia lista de funciones para realizar cálculos entre arreglos [24], [38].
b) *Pandas*. Este paquete es de gran ayuda para leer archivos de bases de datos y los estructura de tal forma que permite trabajar y manipular datos [24].
c) *Matplotlib*. Esta librería es útil para visualizar datos y realizar gráficos en 2D [24].
d) *Scipy*. Es una colección de paquetes con algoritmos eficientes para temas como algebra lineal, representación de matriz dispersa, funciones especiales y funciones estadísticas básicas [24].
e) *Scikit-learn*. Se trata de una librería que ofrece funciones especializadas para las tareas de análisis y minería de datos, como el agrupamiento, la clasificación y la regresión [32].

4.3.2. Lenguaje de programación R

R es un lenguaje de programación que se ha convertido en uno de los más usados para la Ciencia de Datos, así como una herramienta recurrente para las empresas dedicadas al análisis de datos y finanzas como Google, Facebook, Microsoft, *Ford Motors*, *John Deere*, *Lloyds*, entre otras [39].

Este lenguaje es de licencia gratuita y puede ser instalado en plataformas como Windows, Unix/Linux y Mac OS. La estructura de datos básica en R son los vectores, una colección ordenada de valores del mismo tipo. Estos valores pueden ser numéricos, lógicos o de caracter. La lista es otra estructura de datos (vector heterogéneo), así también las matrices y los *dataframes*. Todos los valores de una matriz deben ser del mismo tipo y cada columna debe tener la misma longitud. Los *dataframes* son más flexibles que una matriz, porque pueden presentarse diferentes tipos de valores en distintas columnas [40].

La librería más importante para R, en las tareas de análisis de información, es *tidyverse* [41], la cual contiene los siguientes paquetes:

a) *Ggplot2*: es útil para visualizar datos mediante la creación de gráficos.
b) *Tibble*: maneja una estructura de datos similar al *dataframe*, pero con algunas diferencias, como el despliegue del tipo de dato en cada columna y una función mejorada para imprimir resultados en pantalla.
c) *Tidyr*: se trata de una herramienta para ordenar un conjunto de datos. Considera cada columna como una variable y cada fila como un objeto u observación (con características y valores propios).
d) *Readr*: es un apoyo para la importación sobre varios formatos de bases de datos por ejemplo: csv, tsv, fwf, archivos de Excel, SPSS, Stata, y SAS.
e) *Dplyr*: con este paquete es posible ordenar el despliegue de columnas y filas de un dataset, filtrar datos e incluso crear nuevas variables, a partir de las que ya existen. Además, cuenta con otras funciones para transformar datos. Para profundizar en estas funciones se puede consultar en [41].

Otra de las características importantes de R es que posee una amplia variedad de funciones que se usan como técnicas para análisis de datos. Por ejemplo: modelos lineales, no lineales, pruebas estadísticas clásicas, análisis de series de tiempo, clasificación y agrupación [42].

4.4. Análisis comparativo entre Python y R

4.4.1. Lectura de archivos, precisión y rango de representación numérico

Al leer y procesar diferentes tamaños de archivos, Python y R trabajan en función de la memoria virtual disponible [43, 44]. Este tipo de almacenamiento, es el resultado de la versión del sistema operativo, junto con la versión de la aplicación. Por ejemplo: en una aplicación de 32 *bits* sobre Windows de 64 *bits*, el valor máximo de memoria disponible es menor a 4 Gb, en una aplicación de 64 *bits*, bajo Windows de 64 *bits*, el límite es actualmente de 8 Tb. En Linux, una aplicación de 64 *bits* bajo el sistema operativo de 64 *bits*, puede alcanzar 128 Tb de capacidad.

El manual técnico de Python menciona que la precisión de un número en la parte decimal es de 15 dígitos. Aunque en algunas ocasiones los resultados de los cálculos pueden tener de 15 a 17 dígitos, esto se debe a que dicha precisión esta regularizada con base al formato IEEE-754 [45]. Por ejemplo, la división de 1÷3 da como resultado 0.3333333333333333 con 16 dígitos de precisión decimal, o al dividir 242÷595 se obtiene como resultado 0.40672268907563025 con 17 dígitos de precisión decimal. Asimismo, el rango de representación numérico de Python está entre 1.7976931348623157e+308 a 2.2250738585072014e-308.

Con respecto a R, la presentación de resultados en los cálculos numéricos oscila entre cinco y siete dígitos de precisión decimal. Sin embargo, con el formato adecuado de la función s*printf*, se puede extender esta presentación; de tal forma que se puede observar una precisión de 15 a 17 dígitos en la parte decimal del número, como Python. De acuerdo al manual técnico, la cantidad numérica que R es capaz de representar está entre 2e-308 hasta 2e+308 aproximadamente [46].

Pero, al realizar las pruebas, se observó que el rango de valores exactos es de: 1.797693e+308 a 9.999999e-308.

4.4.2. Comparaciones entre Python y R con respecto a la función del algoritmo K-means

Tanto Python como R tienen integrados dentro de sus paquetes de análisis de datos la función del algoritmo K-means. Por ello, se realizó la tarea de comparar los resultados que se obtuvieron al momento de ejecutar dicha función. El entorno de programación que se utilizó para la ejecución de las instrucciones fue *Spyder* para Python *3.7.1* y *Rstudio* para R 3.4.3. Se trabajó con el sistema operativo *Fedora Scientific*, en la Tabla 7 se describen las características del equipo de cómputo que se utilizó para realizar la experimentación.

Tabla 7 Características del equipo de cómputo

Marca:	Toshiba
Modelo:	Satellite L845
Procesador:	Intel Core i5 Tercera Generación
Velocidad:	2.6 GHz
Memoria RAM:	6 GB

A continuación, en la Tabla 8 se describen las características de 10 bases de datos que se utilizaron para realizar las experimentaciones; estas se descargaron del Centro de Aprendizaje Automático y Sistemas Inteligentes de la Universidad de California, en los Estado Unidos de América. La comunidad científica puede acceder a tal repositorio mediante su página *web* llamada *UCI Machine Learning Repository.* Las bases de datos fueron procesadas de tal manera que únicamente se utilizaron datos numéricos y se eliminaron datos que no correspondían a esta categoría.

Tabla 8 Bases de datos que se utilizaron para la experimentación en la función del algoritmo K-means

Nombre de la base de datos	Número de objetos (n)	Dimensiones (d)
Bank	45211	6
Breastcancer	683	9
Iris	150	4
Poker-hand-testing	1000000	11
Powerplant	9568	5
Rssi	1420	13
Skin_NonSkin	245057	4
Wine	178	13
Household	2049280	4
3D_spatial_network	434874	4

En la Figura 6, se muestra la función K-means que se aplicó para esta experimentación en el lenguaje R. Asimismo, se incluye una breve explicación de los parámetros de entrada. **En el anexo A** se señalan otros parámetros de entrada que pueden aplicarse. Esto significa que la función admite otras combinaciones, de esta manera se da pauta a que en el futuro podrían realizarse más investigaciones de experimentación. Por el momento, la función utilizada en esta investigación fue la que se muestra en la Figura 6.

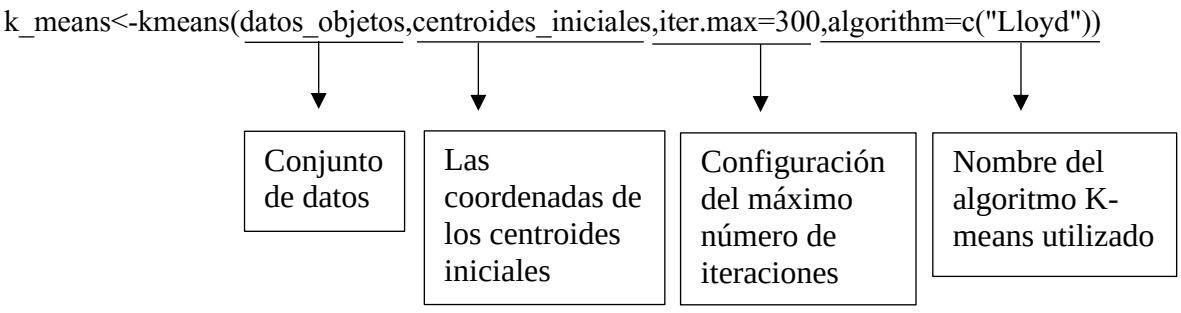

Figura 6 Sintaxis de la función K-means con R

Descripción paramétrica del algoritmo K-means en **R**:

a) Número de iteraciones: como pudo observarse en la Figura 6, el número de iteraciones utilizado fue de 300.
b) Número de grupos: se tomaron en cuenta 10 grupos para todas las bases de datos.

c) Centroides iniciales: los valores de los centroides iniciales fueron tomados aleatoriamente.

En la Figura 7, se muestra la función K-means que se aplicó para la experimentación en el lenguaje Python. Asimismo, **en el anexo B** se mencionan otros parámetros de entrada que pueden aplicarse. Por lo tanto, la función admite otras combinaciones, por lo que es posible que en lo posterior se podrían realizar más investigaciones de experimentación. Por el momento, la función utilizada en esta investigación fue la que se muestra en la Figura 7.

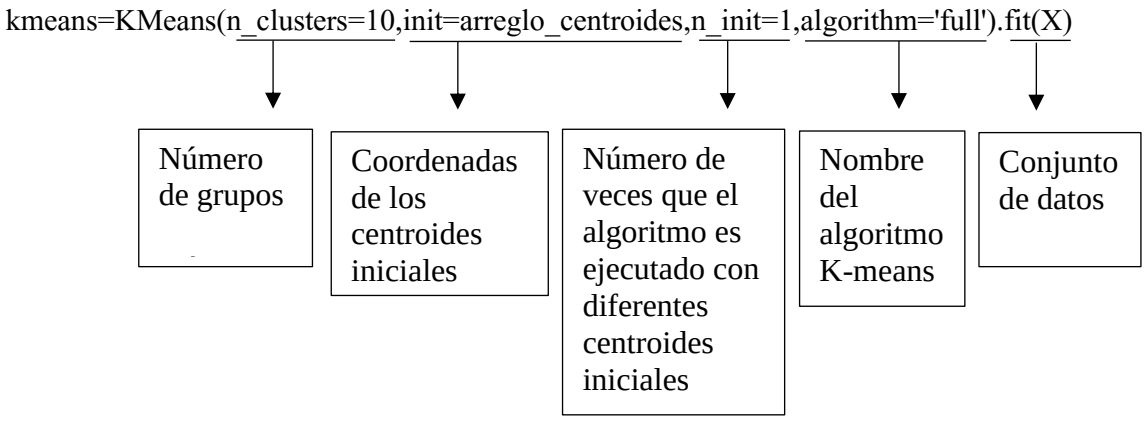

Figura 7 Sintaxis de la función K-means con Python

Descripción paramétrica del algoritmo K-means en **Python**:

a) Número de grupos: se tomaron en cuenta 10 grupos para todas las bases de datos.
b) Centroides iniciales: los valores de los centroides iniciales fueron los mismos que se consideraron para el lenguaje de programación R.
c) Inicialización con diferentes valores de los centroides: la función es capaz de realizar varias ejecuciones con diferentes valores de los centroides, seleccionando la mejor calidad de agrupamiento en el resultado final del error al cuadrado, en este caso como los centroides iniciales con los que se trabajaron fueron dados en un principio, solamente se ejecutó una vez.

En la Tabla 9 se observan los resultados obtenidos al ejecutar K-means con Python y R. En la primera columna se menciona el nombre de la base de datos, la segunda columna señala la cantidad de objetos; la tercera columna menciona el número de dimensiones; la cuarta, quinta y sexta columna señalan el valor del error al cuadrado, tiempo de ejecución y número de iteración, respectivamente, con el lenguaje Python. De la misma manera, pero con el lenguaje R; en la séptima, octava y novena columna se señalan el valor del error al cuadrado, tiempo de ejecución y número de iteración.

Tabla 9 Resultados de ejecución de la función del algoritmo K-means con Python y R

Nombre de la base de datos	Número de objetos	d	Python			R		
			Error al cuadrado	**Tiempo de ejecución**	**It.**	**Error al cuadrado**	**Tiempo de ejecución**	**It.**
Bank	45211	6	21033839410	1.31631613	147	21031778041	0.6272645	163
Breastcancer	683	9	10604.89557	0.00739264	10	10617.94	0.00231004	10
Iris	150	4	27.25381869	0.16377878	5	31.14815	0.00080967	5
Poker-hand-testing	1000000	11	38344710.73	15.7812536	69	38339169	26.31831	200
Powerplant	9568	5	1131003.145	0.05878425	25	1130980	0.03059483	32
Rssi	1420	13	7730979.072	0.00877762	6	7730979	0.00268507	6
Skin_NonSkin	245057	4	292119238.2	0.73522234	9	292077698	0.3374124	23
Wine	178	13	487198.8675	0.00366235	6	487198.9	0.00114727	6
Household	2049280	4	7395989.958	22.3985441	32	7395437	8.735786	79
3D_spatial_network	434874	4	6.42586E+18	2.59436083	26	6.42585E+18	0.9448745	28

En Tabla 10 se muestra un análisis comparativo de los resultados obtenidos. Después de mencionar el nombre de la base de datos, número de objetos y número de dimensiones, en las primeras tres columnas, respectivamente; en las seis columnas restantes que aparecen, señalan de manera correspondiente el lenguaje ganador en tiempo, lenguaje ganador en calidad, porcentaje de pérdida de calidad con R, porcentaje de pérdida de calidad con Python, porcentaje de ganancia de tiempo con R y porcentaje de ganancia de tiempo con Python. Finalmente, cabe

aclarar que en las celdas que muestran guiones significa que en la intersección de la fila y columna no tiene ningún valor, porque carece de sentido alguno.

Tabla 10 Comparación de resultados entre los lenguajes R y Python sobre 10 bases de datos

Nombre de la base de datos	Número de objetos	d	Lenguaje ganador en tiempo	Lenguaje ganador en calidad	% de pérdida de calidad con R	% de pérdida de calidad con Python	% de ganancia de tiempo con R	% de ganancia de tiempo con Python
Bank	45211	6	R	R	----------	0.00980121	52.3469715	----------
Breastcancer	683	9	R	Python	0.12300381	----------	68.752212	----------
Iris	150	4	R	Python	14.2891217	----------	**99.5056322**	----------
Poker-hand-testing	1000000	11	Python	R	----------	0.0144545	----------	**40.0369797**
Powerplant	9568	5	R	R	----------	0.00204643	47.9540322	----------
Rssi	1420	13	R	R	----------	**9.38E-07**	69.4100395	----------
Skin_NonSkin	245057	4	R	R	----------	0.0142223	54.1074337	----------
Wine	178	13	R	Python	0.00000666	----------	68.67392	----------
Household	2049280	4	R	R	----------	0.00747702	60.9984204	----------
3D_spatial_network	434874	4	R	R	----------	**1.71E-05**	63.5796806	----------

El menor porcentaje de pérdida de calidad se obtuvo con Python de 9.38E-07 %, con la base de datos **Rssi** (1420 objetos y 13 dimensiones). El mayor porcentaje en pérdida de calidad se observó con R de 14.2891217 % con la base de datos **Iris** (150 objetos y 4 dimensiones). En el porcentaje de mayor ganancia de tiempo en ejecución se obtuvo con **Iris** de 99.5056322 %; y el menor porcentaje de ganancia en tiempo fue con Python con el 40.0369797 % con la base de datos **Poker-hand-testing** (1000000 objetos y 11 dimensiones).

En la Tabla 11 se muestran los resultados obtenidos al ejecutar K-means con R, sobre 30 ejecuciones de 30 muestras que se tomaron de la base de datos llamada *3D_spatial_ network*. Se configuraron distintos números de grupos y objetos para cada ejecución. En la primera columna se señala el número de la ejecución realizada; en la segunda, el número de objetos utilizados en la muestra; en la tercera columna el número de grupos que se tomaron en cuenta para realizar el agrupamiento; la cuarta columna señala el valor del error al cuadrado; en la quinta se observa el número de iteraciones que realizó la función; y en la última columna se señala el tiempo de ejecución en segundos.

Tabla 11 Resultados de la ejecución del algoritmo K-means aplicando R sobre 30 muestras tomadas de la base de datos 3D_spatial_network

Número de ejecución	Número de objetos	Grupos	Error al cuadrado	Iteraciones	Tiempo de ejecución en segundos
1	6000	5	1.658435e+17	6	0.005808592
2	12000	10	1.96736e+17	8	0.008188248
3	18000	15	6.808541e+16	11	0.04612136
4	24000	20	7.897822e+16	10	0.03564382
5	30000	25	7.051674e+16	8	0.05339932
6	36000	30	8.288985e+16	9	0.06037688
7	42000	35	8.495655e+16	9	0.145164
8	48000	40	1.234762e+17	8	0.1555011
9	54000	45	5.849983e+16	10	0.2054951
10	60000	50	7.179829e+16	9	0.2332542
11	66000	55	6.656479e+16	10	0.2601798
12	72000	60	4.295681e+16	10	0.2404277
13	78000	65	4.134917e+16	8	0.286634
14	84000	70	4.492744e+16	8	0.3142035
15	90000	75	4.216928e+16	9	0.3228884
16	96000	80	3.315263e+16	11	0.5349882
17	102000	85	3.477772e+16	11	0.557802
18	108000	90	3.045399e+16	12	0.6327748
19	114000	95	2.172054e+16	12	0.7556026
20	120000	100	1.912962e+16	12	0.8430903
21	126000	105	1.931471e+16	11	0.7958572
22	132000	110	1.974385e+16	10	0.8626251
23	138000	115	1.995151e+16	12	1.037636
24	144000	120	2.176261e+16	11	0.9988854

25	150000	125	1.919926e+16	11	1.034818
26	156000	130	1.980731e+16	11	1.161413
27	162000	135	1.941815e+16	11	1.348956
28	168000	140	1.55878e+16	12	1.449564
29	174000	145	1.480115e+16	13	1.734416
30	180000	150	1.435112e+16	13	1.816599

En la Tabla 12 se muestran los resultados obtenidos al ejecutar la función del algoritmo K-means con Python. La descripción de las columnas es el mismo al descrito en la Tabla 11. Los valores de los centroides iniciales fueron los mismos que se consideraron al trabajar con R.

Tabla 12 Resultados de ejecución del algoritmo K-means aplicando Python sobre 30 muestras tomadas de la base de datos 3D_spatial_network

Número de ejecución	Número de objetos	Grupos	Error al cuadrado	Iteraciones	Tiempo de ejecución en segundos
1	6000	5	1.6584349149019123e+17	5	0.00961089134216309
2	12000	10	1.9673599660300074e+17	7	0.0239245891571045
3	18000	15	6.808540607609157e+16	10	0.04414534568786621
4	24000	20	7.897821877194429e+16	10	0.07257366180419922
5	30000	25	7.051675495918517e+16	6	0.15265965461730957
6	36000	30	8.208402210376582e+16	9	0.1340475082397461
7	42000	35	8.495654691324994e+16	8	0.15468525886535645
8	48000	40	1.2347624810896962e+17	8	0.15978479385375977
9	54000	45	5.849983445117026e+16	9	0.2446002960205078
10	60000	50	7.179828996314178e+16	8	0.26264333724975586
11	66000	55	6.656479339681939e+16	10	0.3492097854614258
12	72000	60	4.295680914590263e+16	9	0.5012507438659668
13	78000	65	4.13491676405382e+16	8	0.45108938217163086
14	84000	70	4.492792652566602e+16	6	0.43913841247558594
15	90000	75	4.216928155266724e+16	8	0.596235990524292
16	96000	80	3.3152643623760504e+16	9	0.7355115413665771
17	102000	85	3.4777721440598024e+16	10	0.8931810855865479
18	108000	90	3.0543343167412004e+16	8	0.8339533805847168
19	114000	95	2.180885321660807e+16	8	0.933035135269165
20	120000	100	1.920568779745222e+16	9	1.1193382740020752
21	126000	105	1.9317970029919044e+16	11	1.3920142650604248
22	132000	110	1.95249894951027e+16	11	1.5505869388580322
23	138000	115	2.0344339427941384e+16	8	1.317347764968872

24	144000	120	2.1345986016875504e+16	10	1.6901886463165283
25	150000	125	1.9920234293861524e+16	9	1.6963167190551758
26	156000	130	2.009876934032582e+16	9	1.8286361694335938
27	162000	135	1.9142193306798532e+16	9	1.9547042846679688
28	168000	140	1.5243845895774632e+16	9	2.080015182495117
29	174000	145	1.4257363606491282e+16	9	2.22151780128479
30	180000	150	1.2059907288106766e+16	10	2.600801944732666

En la Tabla 13 se realiza una comparación de los resultados entre ambos lenguajes. En la primera columna se presenta el número de comparación con respecto al número de ejecución de ambos lenguajes, la segunda columna señala al lenguaje ganador en tiempo de ejecución; la tercera columna menciona al lenguaje ganador en calidad de agrupamiento; la cuarta, el porcentaje de pérdida de calidad de R; en la quinta, el porcentaje de pérdida de calidad de Python; la sexta columna muestra el porcentaje de ganancia en tiempo de R; y en la última columna se muestra el porcentaje de ganancia en tiempo de Python. Por último, cabe esclarecer que en las celdas que muestran guiones significa que en la intersección de la fila y columna no tiene ningún valor, porque carece de sentido alguno.

Tabla 13 Comparación de resultados entre los lenguajes R y Python de 30 muestras tomadas de la base de datos 3D_spatial_network

Número de ejecución	Lenguaje ganador en tiempo	Lenguaje ganador en calidad	% de pérdida de calidad con R	% de pérdida de calidad con Python	% de ganancia de tiempo con R	% de ganancia de tiempo con Python
1	R	Python	0.000005131	----------	39.56240068	----------
2	R	Python	0.000001727	----------	65.77476024	----------
3	Python	Python	0.000005763	----------	----------	4.284379975
4	R	Python	0.000001555	----------	50.88601138	----------
5	R	R	----------	0.000021214	65.02067286	----------
6	R	Python	0.981711027	----------	54.9585958	----------
7	R	Python	0.000003633	----------	6.155246445	----------
8	R	R	----------	0.000038962	2.680914592	----------
9	R	R	----------	0.000007609	15.98738704	----------
10	R	Python	0.0000000513	----------	11.18975168	----------
11	R	R	----------	0.000005103	25.49469951	----------
12	R	Python	0.000001988	----------	52.03444525	----------
13	R	Python	0.000005706	----------	36.45738265	----------
14	R	R	----------	0.001082914	28.45000777	----------
15	R	R	----------	0.000003682	45.84553681	----------
16	R	R	----------	0.000041094	27.26311283	----------
17	R	R	----------	0.000004142	37.54883427	----------
18	R	R	----------	0.293403811	24.12348043	----------
19	R	R	----------	0.406588495	19.01670458	----------
20	R	R	----------	0.397644059	24.67957904	----------
21	R	R	----------	0.016878482	42.82693648	----------
22	R	Python	1.120925084	----------	44.36783399	----------
23	R	R	----------	1.968920788	21.23294793	----------
24	R	Python	1.951767338	----------	40.90095196	----------
25	R	R	----------	3.75521918	38.99617988	----------
26	R	R	----------	1.471473614	36.48747523	----------
27	R	Python	1.441614808	----------	30.98925446	----------
28	R	Python	2.256347293	----------	30.30993176	----------
29	R	Python	3.814073966	----------	21.92653154	----------
30	R	Python	18.99859308	----------	30.15235152	----------

En esta experimentación se observó que en 29 casos R obtuvo ventaja sobre el tiempo de ejecución. Aunque Python realizó mejor calidad de agrupamiento en 15 ejecuciones. Sin embargo, R hizo mejor calidad de agrupamiento en 15 ejecuciones.

El menor porcentaje de pérdida de calidad se obtuvo con R en la ejecución número 10 con 60,000 objetos y 50 grupos, con un 0.0000000513 % y una ganancia de tiempo de 11.18975168 %. También se obtuvo con R el mayor porcentaje de pérdida de calidad en la ejecución número 30 con 180,000 objetos y 150 grupos. Sin embargo, se alcanzó una ganancia del 30.15235152 % en el tiempo de ejecución. En cuanto al tiempo, se obtuvo en la ejecución número ocho el menor porcentaje de ganancia con 2.680914592 % sobre 48,000 objetos y 40 grupos, con R. En la ejecución número dos se obtuvo un mayor porcentaje de ganancia en tiempo de 65.02067286 % con R y una pérdida de calidad del 0.000001727 %, sobre 12,000 objetos y 10 grupos.

Capítulo 5

Conclusiones y trabajos futuros

No hay nada que temer en la vida, únicamente se debe entender. Ahora es tiempo de entender más, para temer menos.
Marie Curie

5.1. Conclusiones

En esta investigación, se mostró la factibilidad de realizar un análisis comparativo entre lenguajes de programación utilizados para el desarrollo de aplicaciones en el campo de Ciencia de Datos.

Se determinó con base en el estudio y comparación de cinco criterios de selección que se mencionan a continuación, en los siguientes incisos, que Python y R son los lenguajes de programación más eficientes y eficaces para desarrollar aplicaciones en Ciencia de Datos:

a) Ranking de lenguajes más utilizados para desarrollo de aplicaciones científicas, de escritorio y empresarial.
b) Uso de lenguajes por los científicos de datos.
c) Recomendación para el aprendizaje de lenguajes enfocados a Ciencia de Datos.
d) Lenguajes de programación utilizados por las empresas dedicadas a desarrollar aplicaciones en Ciencia de Datos.
e) Clasificación de herramientas de software para el desarrollo de aplicaciones en Ciencia de Datos.

Dentro de esta área de conocimiento, se realizan actividades de colección, exploración, modelado y visualización de la información. Gracias a las funciones especializadas que disponen estos dos lenguajes de programación, es factible desarrollar todas estas actividades.

Derivado del estudio realizado, se puntualizan las siguientes observaciones relevantes:

a) Python cuenta con una representación de resultados numéricos más precisos que R.
b) Ambos lenguajes tienen funciones para importar archivos de datos de cualquier tamaño, su limitante solo es el tamaño de la memoria virtual del sistema operativo.

Por último, con la finalidad de comparar el tiempo de ejecución y la calidad de los resultados de determinadas funciones especializadas, se seleccionó la función K-means. Al experimentar con 10 bases de datos y 30 muestras tomadas de una de las bases de datos llamada *3D_spatial_network* se observó que R presentó un promedio en tiempo de ejecución 1.373487697 segundos y Python 1.737771034 segundos, lo que muestra que R en esta experimentación resultó 1.26 veces más veloz que Python, R mostró una tendencia de ser más veloz que Python, esto se observó debido a que en 38 ejecuciones de 40, obtuvo ventaja en velocidad. En calidad de agrupamiento, para esta experimentación, Python logró en 18 ocasiones mejor calidad y R en 22 ocasiones. En promedio Python obtuvo una pérdida de calidad de 0.208982961 % y R 1.124429758 %.

5.2. Trabajos futuros

Para continuar desarrollando futuras investigaciones, se proponen los siguientes temas:

a) Realizar un análisis comparativo sobre el uso de distintos parámetros en la función K-means de Python y R, con la finalidad de conocer las configuraciones que otorgan eficientes resultados en el dominio de *Big Data*.
b) Mejorar el rendimiento de la función del algoritmo K-means en Python y R utilizando la programación paralela y las versiones compiladas de ambos lenguajes.
c) Un análisis comparativo sobre otra función para el modelado de datos (por ejemplo: regresión logística, regresión lineal, KNN, árboles de clasificación, redes neuronales, entre otros) que Python y R integran en sus paquetes de análisis, utilizando los parámetros de entrada más relevantes.
d) Desarrollar una aplicación en Ciencia de Datos combinando Python y R en un nuevo lenguaje, tal como *PypeR*, *rpy2* o algún otro lenguaje que en el futuro logre la integración de ambos.

Referencias

[1] K. Kambatla, G. Kollias, V. Kumar and A. Grama, “Trends in big data analytics,” *Journal of Parallel and Distributed Computing,* vol. 74, no. 7, July, pp. 2561-2573, 2014.

[2] C. Tsai, C. Lai, H. Chao and A. V. Vasilakos, “Big data analytics: a survey,” *Journal of Big Data,* vol. 2, no. 1, Oct., pp. 1-32, 2015.

[3] N. N. Almanza Ortega, “Desarrollo de heurísticas para la mejora del algoritmo K-Means en las fases de clasificación y convergencia,” PhD. thesis, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, MOR, 2018.

[4] M. Loukides, *What is Data Science?* Sebastopol, CA: O’Reilly Media Inc, 2011.

[5] F. Provost and T. Fawcett, “Data science and its relationship to big data and data-driven decision making,” *Big Data*, vol.1, no. 1, March, pp. 51-59, 2013.

[6] V. Kotu and B. Deshpande. *Data science concepts and practice.* Cambridge, MA: Morgan Kaufmann, 2019.

[7] T. H. Davenport and D. J. Patil, “Data Scientist: The Sexiest Job of the 21st Century,” *Harvard Business Review*, vol. 90, no. 10, Oct., pp. 70-76, 2012.

[8] L. Sánchez Mendoza, “Desarrollo de una aplicación de Ciencia de Datos,” M. S. thesis, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, MOR, 2018.

[9] L. Cao, “Data science: a comprehensive overview,” *ACM Computing Surveys (CSUR)*, vol. 50, no 3, p. 43:1-43:35, 2017.

[10] S. Ozdemir, *Principles of Data Science*, 1st. ed. Birmingham, UK: Pack Publishing Ltd., 2016.

[11] P. Mikalef, M. N. Giannakos, I. O. Pappas and J. Krogstie, "The human side of big data: Understanding the skills of the data scientist in education and industry," In Proc. 2018 IEEE Global Engineering Education Conference (EDUCON), 2018, pp 503-512.

[12] U. O. Lateef, A. Owoade, A. B. L., G. Ogunsanwo, “Introduction to computer programming (Basic),” *researchgate.net,* May 2017. [Online]. Available: https://www.researchgate.net/publication/317182495_INTRODUCTION_TO_COMPUTER_PROGRAMMING_BASIC / [Accessed: Oct. 17, 2019].

[13] D. A. Watt, *Programming Language Design Concepts,* Chichester, WS: John Wiley & Sons Ltd, 2004.

[14] S. Chazallet, *Python3 Los fundamentos del lenguaje,* 2nd ed., Cornellá de Llobregat, BARC: ENI, 2015.

[15] I. Medina, “Programación Estructurada,” *medium.com,* Feb. 19, 2017. [Online]. Available: https://medium.com/laboratoria-how-to/programaci%C3%B3n-estructurada-7fe400bae43d. [Accessed: Jan. 28, 2020].

[16] C. Hughes, T. Hughes. *Parallel and Distributed Programming Using C++*. Reading, MA: Addison Wesley, 2003.

[17] G. Xie and Y. Zhang, “A few of the most popular models for heterogeneous parallel programming,” In Proc. 16th International Symposium on Distributed Computing and Applications to Business*, Engineering and Science*, 2017, pp 15-18.

[18] P. Aalam and T. Siddiqui, “Comparative Study of Data Mining Tools used for Clustering,” In Proc. 3rd International Conference on Computing for Sustainable Global Development, 2016, pp. 3971-3975.

[19] C. Ozgur, T. Colliau, G. Rogers, Z. Hughes and B. Myer-Tyson, “MatLab vs. Python vs. R,” *Journal of Data Science*, vol. 15, no. 3, July, pp. 355-372, 2017.

[20] J. P. Tymoschuk, “Análisis comparativo de eficiencia de lenguajes de programación,” In Proc. 1er Congreso Nacional de Ingeniería Informática/Sistemas de Información, 2013, pp 1-11.

[21] S. Boragan Aruoba and J. Fernández Villaverde, “A comparison of programming languages in macroeconomics,” *Journal of Economic Dynamics and Control*, vol. 58, Sept. pp. 265-273, 2015.

[22] S. Nanz and C. A. Furia, “A Comparative Study of Programming Languages in Rosetta Code,” In Proc. IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015, pp. 778-788.

[23] H. Fangohr, “A comparison of C, Matlab and Python as teaching languages of engineering,” In Proc*.* Internacional Conference on Computational Science*,* 2004, pp. 1210-1217.

[24] T. Siddiqui, M. Alkadri, and N. A. Khan, “Review of Programming Languages and Tools for Big Data Analytics”, *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, May-June, pp. 1112-1118, 2017.

[25] D. Donoho, “50 years of Data Science,” presented at Tukey Centennial Workshop, Princeton, New Jersey, 2015.

[26] R. A. Muenchen, “The Popularity of Data Science Software,” *r4stats.com*, May 28, 2019. [Online]. Available: http://r4stats.com/articles/popularity/ [Accessed: May. 17, 2018].

[27] S. Cass, and P. Bulusu, “Interactive: The Top Programming Languages 2018”. *ieee.org,* July 2018. [Online]. Available: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018 [Accessed: Nov. 15, 2018].

[28] B. Marshall, “Data science experiences for undergraduates,” *Publication:Journal of Computing Sciences in Colleges,* vol. 33, no. 2, Dec, pp. 198-204, 2017.

[29] A. Parameswaran, “Enabling Data Science for the Majority,” In Proc*. VLDB Endowment*, 2019, pp 2309-2322.

[30] S. Kumar, N. Dhanda and A. Pandey, “Data Science — Cosmic Infoset Mining, Modeling and Visualization,” In Proc. 2018 International Conference on Computational and Characterization Techniques in Engineering & Sciences (CCTES), 2018, pp 1-4.

[31] T. Wiktorski, Y. Demchenko and A. Belloum, “Model Curricula for Data Science EDISON Data Science Framework,” In Proc. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017, pp. 369-374.

[32] B. Hayes, “Programming Languages Most Used and Recommended by Data Scientists,” *businessoverbroadway.com*, January 2019. [Online]. Available: http://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/. [Accesed: Feb. 20, 2019].

[33] I. Valchanov, “Who is a typical Data Scientist in 2019?,” kdnuggets.com, March 2019. [Online]. Available: https://www.kdnuggets.com/2019/03/typical-data-scientist-2019.html. [Accesed: Feb. 20, 2019].

[34] G. Piatetsky, “Python eats away at R: Top Software for Analytics, Data Science, Machine Learning in 2018: Trends and Analysis,” *kdnuggets.com*, May 2018. [Online]. Available: https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html. [Accesed: Feb. 21, 2019].

[35] G. V. Rossum, *El tutorial de Python,* Python Software Foundation, 2009. [E-book] Available: http://python.org.ar/pyar/Tutorial. [Accessed: May 15, 2019].

[36] R. G. Duque, “Python para todos,” *mundogeek.net,* October 2008. [Online]. Available: http://mundogeek.net/tutorial-python/. [Accessed: Dic. 01, 2018].

[37] E. L. Suárez, G. S. Herrera and L. M. de la Cruz, “A parallel computing strategy for Monte Carlo simulation using groundwater models,” *Geofísica internacional*, vol. 54, no. 3, Septiembre, pp. 245-254, 2015.

[38] F. Pedregosa et al, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, 12 Oct., pp. 2825−2830, 2011.

[39] P. P. Shinde, K. S. Oza and R. K. Kamat, "Big data predictive analysis: Using R analytical tool," presented at 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2017.

[40] A. Malviya, A. Udhani and S. Soni, "R-tool: Data analytic framework for big data," presented at 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, India, 2016.

[41] H. Wickham and G. Grolemund, *R for Data Science*, Gravenstein Highway North, Sebastopol, CA: O'Reilly Media Inc., 2017.

[42] r-project, "What is R?," [Online]. Available: https://www.r-project.org/about.html. [Accessed: Dic. 07, 2018].

[43] A. Vantol, "Memory Management in Python," realpython.com, November 2018. [Online]. Available: https://realpython.com/python-memory-management/. [Accessed: Sept. 24, 2019].

[44] R Documentation, "Memory Limits in R," stat.ethz.ch, [Online]. Available: https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html. [Accessed: March 15, 2019].

[45] Wextensible, "La precisión del formato IEEE754," wextensible.com, January 2017. [Online]. Available: https://www.wextensible.com/temas/javascript-number/precision.html. [Accesed: Jan. 31, 2020].

[46] R Documentation, "Double-Precision Vectors," stat.ethz.ch, [Online]. Available: https://stat.ethz.ch/R-manual/R-devel/library/base/html/double.html. [Accesed: Oct. 01, 2019].

Anexo A

Sintaxis de la función K-means en R

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)

Parámetro de entrada	**Descripción**
X	Matriz de datos.
centers	Es el número de centroides o una matriz de coordenadas para los centroides iniciales.
iter.max	Número de iteraciones permitidas.
nstart	Funciona cuando el parámetro *centers* contiene un valor numérico. Realiza un determinado número de configuraciones iniciales para los centroides y escoge la mejor configuración.
algorithm	Nombre del algoritmo K-means.
trace	Actualmente es usado para el algoritmo Hartigan-Wong, si su valor es positivo se genera información sobre el progreso del algoritmo.

Anexo B

Sintaxis de la función K-means en Python

KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None, algorithm='auto')

Parámetro de entrada	Descripción
n_clusters	Número de grupos.
init	Se le asigna un método de inicialización, el predeterminado es K-means++. Sin embargo, con el valor 'random' los centroides se escogen aleatoriamente. La otra opción es asignarle una matriz de centroides arbitrariamente.
n_init	Número de veces en el que el algoritmo será ejecutado con diferentes centroides, otorgando la mejor calidad de agrupamiento en el resultado final.
max_iter	Número límite de iteraciones, su valor predeterminado es 300.
tol	Valor de tolerancia en convergencia.
precompute_distances	Es un parámetro que acepta solamente uno de los siguientes tres valores: *auto, true, false*. Cuando el valor es *auto*, este parámetro funciona con la condición de que el número de objetos multiplicado por el número de grupos sea menor o igual a doce millones. Si el *false* no se activa. Si es *true* entonces permite realizar la ejecución de k-means con mayor velocidad pero con mayor consumo de memoria en comparación a que si este parámetro tuviese el valor *false*.
verbose	Acepta un valor del tipo *int* y tiene que ver con la opción de elegir si se desean ver mensajes con relación al desarrollo del trabajo del algoritmo.
random_state	Determina la generación de números aleatorios para la inicialización del centroide.
copy_x	Es un parámetro que predeterminadamente está activado y su función es permitir que los datos originales no se modifiquen.
n_jobs	Indicia el número de procesadores a utilizar en el contexto del cómputo paralelo
algorithm	Se especifica el tipo de algoritmo K-means a utilizar. (*full* o *elkan*).